Uso de servicios OGC

Albert Gavarró Rodríguez

PID_00174758

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción 5

Objetivos 6

1. Google Maps y WMS 7
1.1. Anatomía de un mapa: los *tiles* 7
1.2. Mapas personalizados 8
1.2.1. *Tile layer overlays* 9
1.2.2. Tipos de mapa personalizados 11
1.3. Mapas WMS 14
1.3.1. El servicio OGC/WMS 14
1.3.2. Petición de datos 15
1.3.3. *GTileLayer WMS* 17
1.3.4. Mapas personalizados 21

Resumen 27

Bibliografía 28

Introducción

En los módulos anteriores, hemos aprendido a programar herramientas SIG utilizando siempre la cartografía que acompañaba a la aplicación: en el caso de GoogleMaps, los cuatro tipos de mapa que nos ofrece su API, y en el caso de gvSIG, los mapas que habíamos obtenido previamente de algun servidor de mapas.

Sin embargo, la proliferación de los servicios de cartografía en línea ha hecho posible una nueva forma de trabajar en la que el cliente no dispone ya de una copia propia de los datos, sino que accede continuamente a servicios en línea que mantienen una copia actualizada de esos datos.

En este módulo, aprenderemos a utilizar la cartografía disponible en línea, es decir, a interaccionar con los servicios OGC desde nuestro código para obtener todo tipo de mapas e incorporarlos a nuestra aplicación SIG.

Concretamente, aprenderemos a utilizar un servicio OGC/WMS desde una aplicación web basada en la API de Google Maps, de la misma manera que lo hacen aplicaciones como Wikiloc*. También aprenderemos a combinar las fuentes de datos de Google con los mapas obtenidos mediante el servicio WMS, e incluso prescindiremos de los mapas de Google para combinar dos o más mapas WMS externos.

*** http://www.wikiloc.com. Wikiloc es una aplicación web que permite compartir rutas georreferenciadas.**

Aunque este módulo trata, exclusivamente, sobre el servicio OGC/WMS, las técnicas aquí explicadas son fácilmente trasladables a los demás servicios OGC que proporcionan datos en forma de imágenes.

Objetivos

Al final de este módulo, deberíamos ser capaces de:

1. Crear tipos de mapa personalizados en Google Maps.
2. Agregar nuevos tipos de mapa a Google Maps.
3. Llamar a un servicio OGC/WMS desde una aplicación web.
4. Crear mapas personalizados en GoogleMaps que incorporen datos procedentes de servicios WMS.
5. Combinar los mapas de Google Maps con mapas WMS externos.
6. Combinar dos o más fuentes WMS prescindiendo de los mapas de Google Maps.

1. Google Maps y WMS

1.1. Anatomía de un mapa: los *tiles*

Todos los mapas de Google Maps estan divididos en porciones de 256 × 256 píxeles llamadas *tiles*. Cuando se solicita un mapa, Google Maps descarga del servidor las distintas porciones por separado y las junta, creando la ilusión de una única imagen:

El objetivo de reducir el mapa en porciones no es otro que minimizar el tamaño de los datos que el servidor tiene que enviar a las distintas aplicaciones. Debemos pensar que si bien en niveles de *zoom* pequeños sí que sería factible tener todo el mapa en una única imagen (cuyo tamaño sería de alrededor de unos pocos kilobytes), no sucede lo mismo con los niveles de *zoom* grandes, en los que sería prácticamente imposible enviar el mapa entero (cuyo tamaño sería de gigabytes).

En el nivel de *zoom* más pequeño (el nivel 0), el planeta entero está representado en una única porción. Cada nivel subsiguiente de *zoom* divide cada porción en cuatro subporciones, de manera que el número total de porciones responde a la fórmula 4^N, donde N es el nivel de *zoom*. Así pues, en el nivel 1 habrá cuatro porciones (que formarán una rejilla de 2 × 2), en el nivel 2, dieciséis (que formarán una rejilla de 4 × 4), y así sucesivamente.

Dependiendo del tamaño del *viewport*, la coordenada central del mapa y el nivel de *zoom*, se mostrarán unas porciones u otras.

1.2. Mapas personalizados

La API de Google Maps nos permite definir mapas personalizados, cuyo contenido (el contenido de cada una de las porciones) podemos elegir arbitrariamente. Una vez creados, estos mapas se pueden agregar a la vista mediante dos mecanismos diferenciados:

- como una capa del mapa actual, o
- como un mapa independiente.

Aunque existan dos formas de agregar mapas personalizados a una vista, la forma de implementarlos es siempre la misma: crear un objeto "GTileLayer" y sobrecargar las funciones siguientes:

- ***getTileUrl***. Devuelve la dirección (URL) que contiene la imagen de la porción (*tile*) del mapa, dadas las coordenadas de la porción y el nivel de *zoom*.

- ***isPng***. Devuelve un valor booleano indicando si la imagen de la porción está en formato PNG*, en cuyo caso puede contener transparencias.

- ***getOpacity***. Devuelve un valor entre 0,0 y 1,0 indicando el nivel de opacidad del mapa. 0,0 es completamente transparente, y 1,0, completamente opaco.

*** PNG son las siglas de *Portable Network Graphics* (gráficos de red portables). El formato PNG es un formato gráfico basado en un algoritmo de compresión sin pérdida no sujeto a patentes. http://tools.ietf.org/html/rfc2083**

Ejemplo 1: añadir una marca de agua a un mapa

Veamos un ejemplo sencillo de implementación de un mapa personalizado. Supongamos que deseamos mostrar los mapas de Google con el logotipo de la UOC sobreimpreso a modo de marca de agua*.

*** Una marca de agua es una imagen translúcida que se muestra sobre otra imagen o texto y que sirve para identificar el origen del documento.**

La idea es que la marca de agua se vaya repitiendo a lo largo y ancho del mapa, a intervalos regulares. Para lograr este objetivo, emplazaremos una misma imagen —el logotipo de la UOC— en todas y cada una de las porciones que, recordemos, dividen el mapa en áreas de 256 × 256 píxeles. A todos los efectos, es como si creásemos un mosaico con el logotipo de la UOC. Para crear el efecto de marca de agua, estableceremos el nivel de opacidad del mapa en el 50%.

Suponiendo que tenemos el logotipo de la UOC en un fichero llamado "logo-uoc.png" (de 256 × 256 píxeles), el código que crea el mapa nos quedaría así:

```
var capaLogo;

capaLogo = new GTileLayer(null, 0, 19);

// el nombre del fichero que contiene la imagen
capaLogo.getTileUrl = function() { return "logo-uoc.png"; }

// opacidad del 50%, expresada en tanto por uno
capaLogo.getOpacity = function() { return 0.5; }

// el fichero que contiene la imagen está en formato PNG
capaLogo.isPng = function() { return true; }
```

Como podemos observar, el código sobrecarga las tres funciones de la clase "GTileLayer" antes descritas: *getTileUrl*, *getOpacity* e *isPng*. La función *getTileUrl* siempre devuelve la misma imagen ("logo-uoc.png"), que es la que queremos mostrar en todas las porciones del mapa; *getOpacity*, siempre la misma transparencia (del 50%), e *isPng*, siempre *true*, puesto que sabemos que la imagen que se devuelve está en formato PNG. Por otro lado, el constructor de la clase "GTileLayer" espera tres parámetros:

- una colección de textos de *copyright* que se mostrarán en la parte inferior derecha del mapa,
- el nivel de *zoom* mínimo del mapa (ampliación mínima del mapa) y
- el nivel de *zoom* máximo del mapa (ampliación máxima del mapa).

En el ejemplo, no se ha proporcionado ninguna colección de textos de *copyright* ("null") y se han establecido los niveles de *zoom* mínimo y máximo permitidos por Google Maps.

En los apartados siguientes, aprenderemos a agregar este mapa a la vista.

1.2.1. *Tile layer overlays*

Una forma de agregar un mapa personalizado a la vista es mediante el mecanismo de las capas. Google llama a las capas que contienen mapas *tile layer overlays*.

Por medio de este mecanismo, situaremos el mapa en la capa más superficial de la vista, de manera que quede por encima de las demás. Dada su posición predominante, debemos jugar con su opacidad para que revele, en mayor o menor medida, las capas inferiores.

Los pasos que se habrán de seguir serán dos:

1) crear una capa a partir del objeto "GTileLayer" que contiene el mapa, y
2) agregar dicha capa a la vista.

Siguiendo con el ejemplo anterior, el código siguiente:

```
mapa.addOverlay(new GTileLayerOverlay(capaLogo));
```

crea una capa a partir del mapa “capaLogo” y lo agrega a la vista del mapa.

Como se puede observar, el primer paso se realiza instanciando un objeto "GTileLayerOverlay", cuyo constructor espera un objeto "GTileLayer", es decir, el mapa personalizado ("capaLogo"). El segundo paso se alcanza mediante una llamada a la función *addOverlay* de la clase "GMap2", cuyo único parámetro debe ser un objeto "GTileLayerOverlay", es decir, la capa que se desea agregar a la vista.

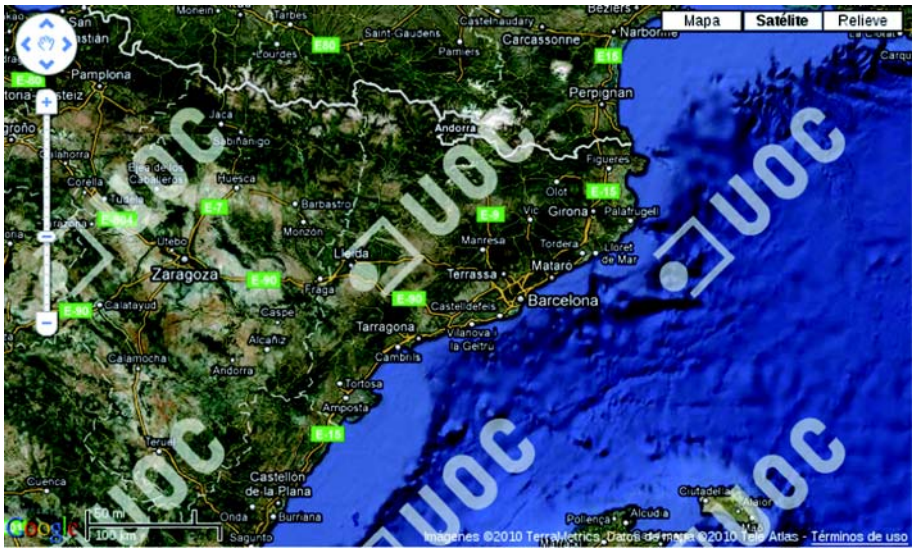
Debemos notar que estamos agregando el mapa a la vista, del mismo modo en que se agrega un marcador. En consecuencia, tanto el mapa como el marcador se mostrarán en el primer plano de la vista.

Juntándolo todo, el código nos quedaría así:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
  content="text/html;charset=utf-8"/>
<title>Ejemplo GTileLayer</title>
<script
  src="http://maps.google.com/maps?file=api&amp;v=2"
  type="text/javascript">
</script>
<script type="text/javascript">
function InicializarMapa() {
  var mapa;
  var capaLogo;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    mapa.setCenter(new GLatLng(41.5580, 1.5906), 7);
    mapa.setUIToDefault();

    capaLogo = new GTileLayer(null, 0, 19);

    // el nombre del fichero que contiene la imagen
    capaLogo.getTileUrl = function() {
```

```
      return "logo-uoc.png";
    }
    // opacidad del 50%, expresada en tanto por uno
    capaLogo.getOpacity = function() { return 0.5; }

    // el fichero que contiene la imagen está en formato PNG
    capaLogo.isPng = function() { return true; }

    mapa.addOverlay(new GTileLayerOverlay(capaLogo));
  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
  style="width: 750px; height: 450px"></div>
</body>
</html>
```

El resultado será el siguiente:



1.2.2. Tipos de mapa personalizados

Otra forma de agregar un mapa personalizado a una vista es creando un tipo de mapa personalizado. Mediante este mecanismo, mostraremos el nuevo mapa aislado de los demás y podremos acceder a él mediante un botón dedicado que se mostrará al lado de los botones de los tipos de mapa predeterminados.

Los pasos que se habrán de seguir serán dos:

1) crear un tipo de mapa personalizado, definiendo el contenido de sus capas, y
2) agregar el nuevo tipo de mapa a la vista.

Partiendo otra vez del mapa que hemos creado con el mosaico de logotipos de la UOC, el código siguiente:

```
var mapaLogo;

mapaLogo = new GMapType([capaLogo],
   G_NORMAL_MAP.getProjection(), "Marca de agua");
mapa.addMapType(mapaLogo);
```

crea un nuevo tipo de mapa (llamado "Marca de agua") y lo agrega a la vista del mapa.

Como se puede observar, el primer paso se realiza instanciando un objeto de la clase "GMapType", cuyo constructor espera tres parámetros: un vector con las capas que contiene el mapa, la proyección del mapa y el nombre del tipo de mapa. Debemos observar que el tipo de mapa que acabamos de crear sólo tendrá una capa —la del mosaico del logotipo de la UOC, "capaLogo"—, y tendrá la misma proyección que el mapa normal ("G_NORMAL_MAP"), que obtenemos mediante la función *getProjection*.

El segundo paso se realiza mediante la función *addMapType* de la clase "GMap2", que espera un objeto "GMapType".

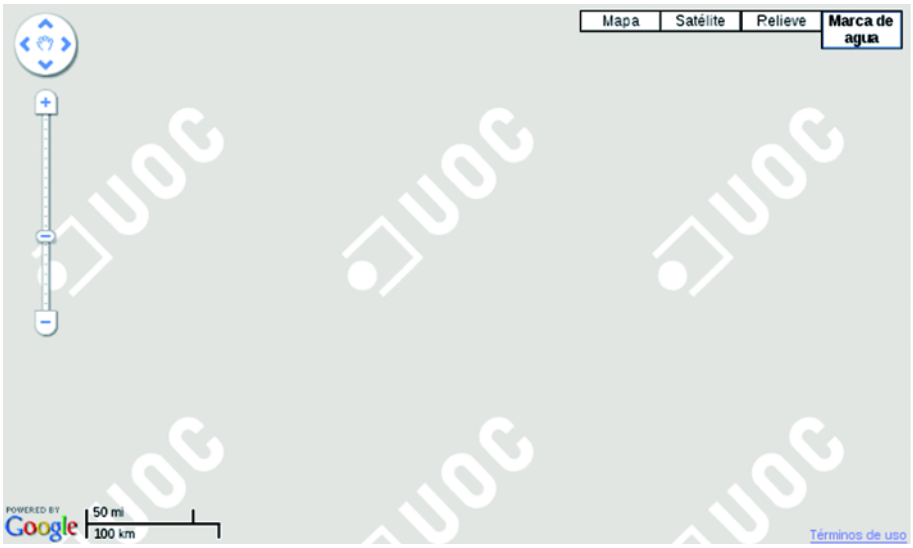
Juntándolo todo, el código nos quedaría así:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
  content="text/html;charset=utf-8"/>
<title>Ejemplo GTileLayer</title>
<script
  src="http://maps.google.com/maps?file=api&amp;v=2"
  type="text/javascript">
</script>
<script type="text/javascript">
function InicializarMapa() {
  var mapa;
  var capaLogo;
  var mapaLogo;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    mapa.setCenter(new GLatLng(41.5580, 1.5906), 7);
```

```
    mapa.setUIToDefault();

    capaLogo = new GTileLayer(null, 0, 19);

    // el nombre del fichero que contiene la imagen
    capaLogo.getTileUrl = function() {
      return "logo-uoc.png";
    }

    // opacidad del 100%, expresada en tanto por uno
    capaLogo.getOpacity = function() { return 1.0; }

    // el fichero que contiene la imagen está en formato PNG
    capaLogo.isPng = function() { return true; }

    // crea un nuevo tipo de mapa
    mapaLogo = new GMapType([capaLogo],
      G_NORMAL_MAP.getProjection(),
      "Marca de agua");
    mapa.addMapType(mapaLogo);
  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
  style="width: 750px; height: 450px"></div>
</body>
</html>
```

Como en esta ocasión la capa que contiene el mosaico se muestra aislada, hemos cambiado su opacidad a 1,0, de manera que sea completamente opaca.

El resultado será el siguiente:



1.3. Mapas WMS

Hasta ahora hemos aprendido a crear mapas personalizados y a agregarlos, como capas o como mapas aislados, a la vista de mapa. Sin embargo, el contenido que mostrábamos era muy poco interesante: tan sólo una imagen que se repetía una y otra vez.

Ahora utilizaremos esta base para crear mapas personalizados que muestren datos obtenidos de un servicio OGC/WMS. La idea que reside en el fondo de esta técnica es relativamente simple: obtener la imagen de cada porción de un servicio OGC/WMS y proporcionarla a Google Maps.

1.3.1. El servicio OGC/WMS

El servicio OGC/WMS, en adelante WMS (por *Web Map Service,* servicio de mapas de la Web), proporciona mapas georeferenciados a partir de datos geográficos. Los mapas se generan a petición de un cliente, que especifica, mediante un protocolo estándar*, los datos que quiere obtener, la zona geográfica en la que se limita la petición, el formato del mapa obtenido, etc. Al final, el cliente obtiene una imagen georeferenciada que concentra y representa toda la información que esperaba obtener. Es importante destacar que el cliente no recibe nunca los datos en sí: tan sólo una imagen apta (hablando en términos de resolución) para ser mostrada en la pantalla de un ordenador.

*** Se puede consultar la especificación del protocolo en: http://www.opengeospatial.org/standards/wms**

El estándar define dos operaciones obligatorias que debe soportar el servicio (*GetCapabilities* y *GetMap*) y otra optativa (*GetFeatureInfo*):

- ***GetCapabilities***. Proporciona información acerca de los datos ofrecidos por el servidor y sus características, los sistemas de referencia soportados, el ámbito geográfico, etc.

- ***GetMap***. Proporciona una imagen georeferenciada que contiene los datos solicitados. El estándar nos permite definir el área del mapa, las capas mostradas, el formato de la imagen, etc.

- ***GetFeatureInfo***. Proporciona información acerca de características particulares mostradas en el mapa.

La información que se puede obtener por medio del servicio WMS gira en torno a las capas de información. Es decir, el proveedor proporciona una o más capas, que se pueden obtener aisladas o combinadas mediante la operación *GetMap*. Es importante remarcar que siempre se obtiene una única imagen como respuesta a una operación *GetMap*.

A continuación, se detallan los parámetros obligatorios que espera la operación *GetMap* y el significado de cada uno de ellos:

- **VERSION=1.1.1.** Especifica el número de versión del protocolo. En otras palabras, especifica la versión del servicio a la que se espera acceder. Dependiendo de la versión, el proveedor aceptará unos parámetros u otros. Estos materiales se basan exclusivamente en la versión 1.1.1 del estándar.
- **REQUEST=GetMap.** Especifica la operación a la que se quiere acceder. Para la operación *GetMap*, el valor de REQUEST siempre será *GetMap*.
- **LAYERS (capas).** Especifica las capas de información que se desean obtener, separadas por comas. Las capas se deben listar desde la más profunda hasta la más superficial.
- **STYLES (estilos).** Especifica el estilo con el que se dibujará cada capa. Igual que en el parámetro anterior, los nombres de los estilos deben ir separados por comas.
- **SRS (de *Source Reference System*, 'sistema de referencia de la fuente').** Especifica el sistema de referencia de la petición. Todas las coordenadas especificadas en los demás parámetros se suponen escritas en este sistema de referencia.
- **BBOX (de *Bounding BOX*, 'cuadro delimitador').** Especifica los límites del mapa que se desea obtener. Se trata de una lista separada por comas de las coordenadas suroeste y noreste del mapa. Las coordenadas deben especificarse de acuerdo al sistema de referencia indicado en el parámetro SRS.
- **WIDTH (ancho).** Especifica la anchura, medida en píxeles, que debe tener la imagen resultante.
- **HEIGHT (alto).** Especifica la altura, medida en píxeles, que debe tener la imagen resultante.
- **FORMAT (formato).** Especifica el formato (PNG, JPEF, GIF, etc.) que debe tener la imagen resultante.

En resumen, mediante la operación *GetMap* podemos obtener una porción de cualquier mapa que nos ofrezca el proveedor, en el formato y en las dimensiones que deseemos.

1.3.2. Petición de datos

Toda la potencia y la flexibilidad que nos ofrece el servicio WMS queda ensalzada por otra característica no menos importante del servicio: su accesibilidad desde la Web. Como su nombre indica (*Web Map Service*), cualquier cliente de la Web puede conectarse a un servicio WMS y obtener mapas.

En este sentido, el estándar ha dispuesto que uno de los métodos de acceso a las operaciones WMS sea uno de los mecanismos fundamentales de la Web: el método HTTP GET. En otras palabras, se puede acceder a las operaciones mediante una URL cualquiera, de la misma forma que solicitamos una página web.

Así pues, si introducimos la URL siguiente en nuestro navegador:

```
http://shagrat.icc.es/lizardtech/iserv/
ows?SERVICE=WMS&REQUEST=GetMap&VERSION=1.1.1&LAYERS=mt
c50m&STYLES=&SRS=EPSG:23031&BBOX=290368.84,4538236.42,
292203.28,4540070.86&WIDTH=500&HEIGHT=500&FORMAT=JPEG
```

obtendremos un mapa topográfico a escala 1:50000 de Benifallet y sus alrededores.

Si observamos detenidamente la URL, veremos los parámetros que hemos descrito en el apartado anterior separados por el carácter *et* ("&", *ampersand* en inglés): REQUEST, SERVICE, VERSION, SRS, BBOX, WIDTH, HEIGHT, LAYERS, STYLES y FORMAT. La tabla siguiente muestra los valores asignados a cada uno de ellos y su significado:

Parámetro	Valor	Significado
SERVICE	WMS	Especifica que se desea acceder al servicio WMS.
REQUEST	GetMap	Especifica que se desea realizar la operación *GetMap*.
VERSION	1.1.1	Especifica que se desea utilizar la versión 1.1.1 del protocolo WMS.
LAYERS	mtc50m	Especifica que se desea acceder a los datos de la capa "mtc50m". Las capas serán exclusivas de cada servidor, y se podrá obtener una lista de ellas mediante la operación *GetCapabilities*. En este caso, se trata de un mapa topográfico a escala 1:50000 ofrecido por el ICC.
STYLES	vacío	Especifica que no se desea ningún estilo en particular. De hecho, la capa seleccionada no tiene ningún estilo asociado. De nuevo, los estilos son propios de cada servidor y de cada capa. Se puede obtener una lista de estilos mediante la operación *GetCapabilities*.
SRS	EPSG:23031	Especifica que las coordenadas utilizadas en los demás parámetros estarán escritas según el sistema de referencia European Datum 1950 proyección UTM Huso 31 Norte (EPSG:23031).
BBOX	290368.84, 4538236.42, 292203.28, 4540070.86	Especifica los límites del área que se desea obtener mediante las coordenadas de la esquina superior derecha (290368.84, 4538236.42) y la esquina inferior izquierda (292203.28, 4540070.86) del mapa. Recordemos que estas coordenadas están escritas en EPSG:23031.
WIDTH	500	Especifica que se desea obtener una imagen cuya anchura sea de 500 píxeles.
HEIGHT	500	Especifica que se desea obtener una imagen cuya altura sea de 500 píxeles.
FORMAT	JPEG	Especifica que se desea obtener una imagen en formato JPEG.

1.3.3. *GTileLayer WMS*

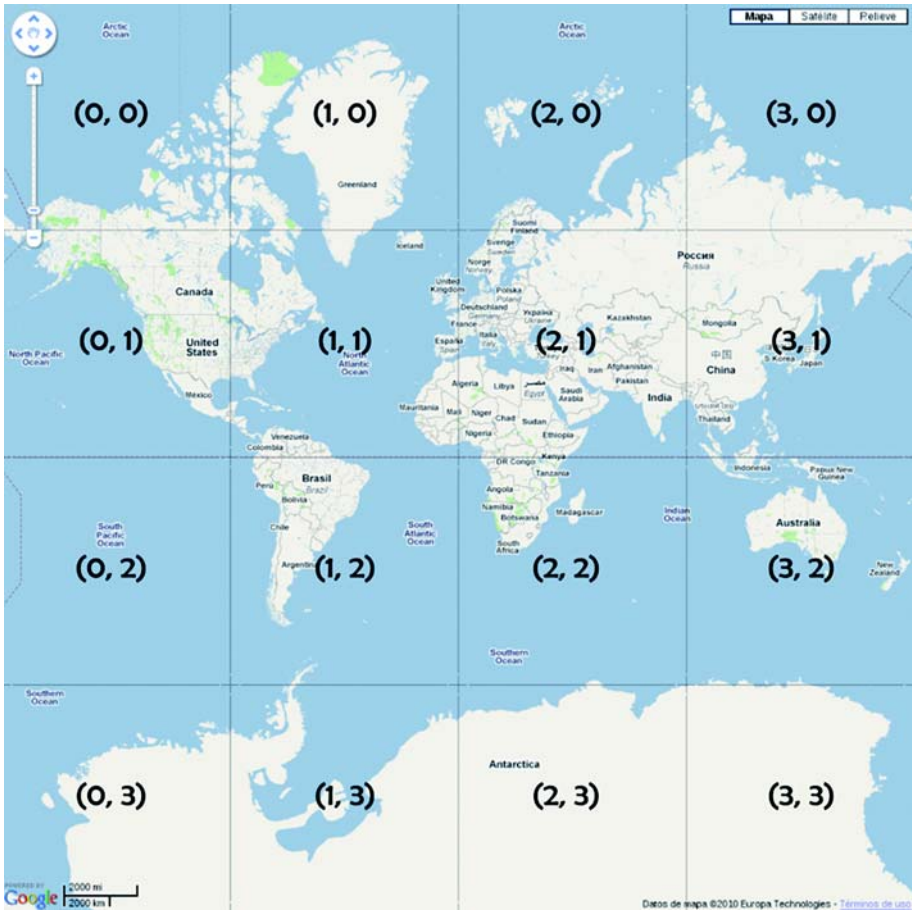
Ahora que ya sabemos cómo obtener una porción de un mapa mediante un servicio WMS, veamos cómo podemos trasladar estas porciones a un mapa personalizado de Google Maps.

Como ya hemos visto, un mapa de Google Maps se divide en porciones cuadradas de 256×256 píxeles. Además, el número de porciones varía en función del *zoom*, y obedece a la fórmula 4^N, donde N es el nivel de *zoom*. Así, en el nivel de *zoom* 0 (menor ampliación), tenemos una única porción; en el nivel 1, cuatro (formando una rejilla de 2×2); en el nivel 2, dieciséis (formando una rejilla de 4×4), y así sucesivamente.

Consecuentemente, podemos decir que en un mapa existen dos tipos de coordenadas:

- las coordenadas de un píxel dentro de una porción, y
- las coordenadas de una porción dentro del mapa.

Ambos tipos de coordenadas se expresan mediante dos componentes, x e y, donde x es el número de columna en la que se encuentra el píxel o la porción dentro de la rejilla, e y es el número de fila. En ambos casos, se empieza contando desde 0.



Cuando nos hemos introducido en la creación de mapas personalizados, hemos mencionado que la función *getTileUrl* de la clase "GtileLayer" recibe por parámetro las coordenadas de la porción y el nivel de *zoom*. Aunque entonces no ha sido necesario consultar ninguno de sus parámetros porque siempre se mostraba la misma imagen en todas las porciones, ahora ello sí que será necesario, puesto que el contenido de una porción u otra variará en función de la parte del mapa que se represente.

Para representar correctamente la parte del mapa correspondiente a una porción, debemos determinar el área que cubre dicha porción. Aquí ya empezamos a tener un punto de contacto con la función *GetMap* del servicio WMS, cuyo parámetro BBOX especificaba el área del mapa que se quería obtener mediante las coordenadas de las esquinas noreste y suroeste. Si somos capaces de obtener dichas coordenadas de una porción, podremos obtener la imagen del mapa por medio del servicio WMS.

Trasladadas al sistema de coordenadas de la API de Google Maps, las coordenadas noreste y sureste corresponden a las coordenadas de las esquinas superior izquierda e inferior derecha. Para obtenerlas, seguiremos los pasos siguientes:

1) calcular las coordenadas en píxeles de las esquinas de la porción, y
2) convertir las coordenadas en píxeles a coordenadas geográficas.

El primer paso lo resolveremos con unas pocas multiplicaciones y sumas. Dada una porción situada en la posición (x, y) de una rejilla isométrica en la que cada porción mide 256 × 256 píxeles, las coordenadas (v, w), expresadas en píxeles, de sus esquinas superior izquierda e inferior derecha responden a las fórmulas:

	Superior izquierda	**Inferior derecha**
v	$x * 256$	$(x + 1) * 256$
w	$(y + 1) * 256$	$y * 256$

El segundo paso lo realizaremos mediante la función *fromPixelToLatLng*, de la clase "GProjection". El objeto "GProjection" (hay un objeto "GProjection" asociado a cada mapa), que gestiona la transformación de coordenadas, lo obtendremos mediante la función *getProjection* de la clase "GMap2".

Recapitulando, dado un objeto llamado "tile" de tipo "GPoint" que representa las coordenadas (x, y) de una porción del mapa, y un nivel de *zoom* representado por la variable "zoom", las líneas siguientes calculan las coordenadas geográficas que debemos pasar al servicio WMS ("geoSupIzq" y "geoInfDer", por este orden) para obtener la imagen del mapa que debe mostrarse en la porción representada por el objeto "tile".

```
// obtiene las coordenadas x/y de las esquinas superior
// izquierda e inferior derecha
xySupIzq = new GPoint(tile.x * 256, (tile.y + 1) * 256);
xyInfDer = new GPoint((tile.x + 1) * 256, tile.y * 256);

// obtiene las coordenadas geográficas de las esquinas
// superior izquierda e inferior derecha
geoSupIzq = G_NORMAL_MAP.getProjection().
  fromPixelToLatLng(xySupIzq, zoom);
geoInfDer = G_NORMAL_MAP.getProjection().
  fromPixelToLatLng(xyInfDer, zoom);
```

Como se puede observar, la función *fromPixelToLatLng* transforma las coordenadas expresadas en píxeles en coordenadas geográficas. Dado que el número de porciones y, consecuentemente, las dimensiones del mapa varían con el nivel de *zoom*, la función *fromPixelToLatLng* necesita, además, que se le proporcione el nivel de *zoom* actual para poder realizar la transformación correctamente.

Ahora sólo nos queda construir la llamada a la función *GetMap* del servicio WMS con las coordenadas obtenidas. Las líneas siguientes:

```
// genera la URL del "tile"
urlTile  = wmsUrl;
urlTile += "&REQUEST=GetMap";
urlTile += "&SERVICE=WMS";
urlTile += "&VERSION=" + wmsVersion;
urlTile += "&LAYERS=" + wmsCapas;
urlTile += "&STYLES=" + wmsEstilos;
urlTile += "&FORMAT=" + wmsFormato;
urlTile += "&BGCOLOR=" + wmsColorFondo;
urlTile += "&TRANSPARENT=TRUE";
urlTile += "&SRS=" + this.wmsSrs;
urlTile += "&BBOX=" + geoSupIzq.x + "," + geoSupIzq.y +
  "," + geoInfDer.x + "," + geoInfDer.y;
urlTile += "&WIDTH=256";
urlTile += "&HEIGHT=256";
urlTile += "&reaspect=false";
```

almacenan en la variable “urlTile” la URL que nos proporcionará la imagen de la porción caracterizada por las coordenadas “geoSupIzq” y “geoInfDer”. Las variables “wmsUrl” (la dirección base del servidor), “wmsVersion”, “wmsCapas”, “wmsEstilos”, “wmsFormato” y “wmsColorFondo” podrán contener valores distintos según el servidor al que nos hayamos conectado y las necesidades que tengamos. El significado de cada uno de estos parámetros se ha explicado en el apartado “El servicio OGC/WMS”.

Juntándolo todo, la función *getTileUrl* de un "GTileLayer" ligado a un servicio WMS tendrá un aspecto parecido al siguiente:

```
// tile: GPoint
// zoom: Number
function obtenerUrlTile(tile, zoom) {
  // coordenada x/y de la esquina superior izquierda
  var xySupIzq;
  // coordenada geográfica de la esquina superior izquierda
  var geoSupIzq;
  // coordenada x/y de la esquina inferior derecha
  var xyInfDer;
  // coordenada geográfica de la esquina inferior derecha
  var geoInfDer;
  var urlTile;

  // obtiene las coordenadas x/y de las esquinas superior
  // izquierda e inferior derecha
  xySupIzq = new GPoint(tile.x * 256, (tile.y + 1) * 256);
  xyInfDer = new GPoint((tile.x + 1) * 256, tile.y * 256);

  // obtiene las coordenadas geográficas de las esquinas
  // superior izquierda e inferior derecha
  geoSupIzq = G_NORMAL_MAP.getProjection().
    fromPixelToLatLng(xySupIzq, zoom);
  geoInfDer = G_NORMAL_MAP.getProjection().
    fromPixelToLatLng(xyInfDer, zoom);

  // genera la URL del "tile"
  urlTile  = this.wmsUrl;
  urlTile += "&REQUEST=GetMap";
  urlTile += "&SERVICE=WMS";
  urlTile += "&VERSION=" + this.wmsVersion;
  urlTile += "&LAYERS=" + this.wmsCapas;
  urlTile += "&STYLES=" + this.wmsEstilos;
  urlTile += "&FORMAT=" + this.wmsFormato;
  urlTile += "&BGCOLOR=" + this.wmsColorFondo;
  urlTile += "&TRANSPARENT=TRUE";
  urlTile += "&SRS=" + this.wmsSrs;
  urlTile += "&BBOX=" + geoSupIzq.x + "," + geoSupIzq.y +
    "," + geoInfDer.x + "," + geoInfDer.y;
  urlTile += "&WIDTH=256";
  urlTile += "&HEIGHT=256";
  urlTile += "&reaspect=false";

  return urlTile;
}
```

1.3.4. Mapas personalizados

Partiendo del código desarrollado en el apartado anterior, a continuación mostraremos algunos ejemplos que combinan los mapas de Google con mapas WMS, o que incluso prescinden de los mapas de Google para mostrar un único mapa WMS o un par de ellos combinados. Para ilustrar estos ejemplos, hemos utilizado los servicios WMS que ofrecen el ICC* i el CREAF**.

*** Los datos proporcionados por el servicio WMS del ICC se detallan en la página:**
http://www.icc.es/web/content/es/prof/cartografia/fitxes_geoserveis.html
**** Los datos proporcionados por el servicio WMS del CREAF se detallan en la página:**
http://www.opengis.uab.es/

Ejemplo 2: un único mapa WMS

En este ejemplo, se crea un mapa personalizado que contiene un único mapa topográfico a escala 1:250000 procedente del servicio WMS del ICC:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
  content="text/html;charset=utf-8"/>
<title>Ejemplo 2</title>
<script src="http://maps.google.com/maps?file=api&amp;v=2"
  type="text/javascript"></script>
<script type="text/javascript">
function initialize() {
  var mapa;
  var capaTopo;
  var mapaTopo;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(document.getElementById("elemento_mapa"));

    mapa.setCenter(new GLatLng(41.5580, 1.5906), 7);
    mapa.setUIToDefault();

    // crea la capa WMS
    capaTopo = crearCapaWms(7, 12, 1.0,
      "http://shagrat.icc.es/lizardtech/iserv/ows?",
      "mtc250m,");

    // crea un mapa personalizado a partir de la capa WMS
    mapaTopo = new GMapType(
      [capaTopo],
      G_NORMAL_MAP.getProjection(),
      "WMS");

    // agrega el mapa personalizado al visor
    mapa.addMapType(mapaTopo);
  }
}

function crearCapaWms(zoomMin, zoomMax, opacidad, url, capas,
  estilos, formato, colorFondo, version, srs) {
  var capaWms;

  capaWms = new GTileLayer(null, zoomMin, zoomMax);

  // rellena los parámetros no especificados
  if (! estilos) { estilos = ""; }
  if (! formato) { formato = "image/png"; }
  if (! version) { version = "1.1.1"; }
  if (! colorFondo) { colorFondo = "0xFFFFFF"; }
  if (! srs) { srs = "EPSG:4326"; }
```

```
  capaWms.wmsUrl = url;
  capaWms.wmsCapas = capas;
  capaWms.wmsEstilos = estilos;
  capaWms.wmsFormato = formato;
  capaWms.wmsVersion = version;
  capaWms.wmsColorFondo = colorFondo;
  capaWms.wmsSrs = srs;

  // la dirección web del fichero que contiene la imagen
  capaWms.getTileUrl = obtenerUrlTile;

  // opacidad expresada en tanto por uno
  capaWms.getOpacity = function() { return opacidad; }

  // el fichero que contiene la imagen está en formato PNG?
  capaWms.isPng = function() { return formato == "image/png"; }

  return capaWms;
}

// tile: GPoint
// zoom: Number
function obtenerUrlTile(tile, zoom) {
  var xySupIzq;  // coord. x/y de la esquina superior izquierda
  var geoSupIzq; // coord. geográfica de la esquina superior
                 // izquierda
  var xyInfDer;  // coord. x/y de la esquina inferior derecha
  var geoInfDer; // coord. geográfica de la esquina inferior
                 // derecha
  var urlTile;

  // obtiene las coordenadas x/y de las esquinas superior
  // izquierda e inferior derecha
  xySupIzq = new GPoint(tile.x * 256, (tile.y + 1) * 256);
  xyInfDer = new GPoint((tile.x + 1) * 256, tile.y * 256);

  // obtiene las coordenadas geográficas de las esquinas superior
  // izquierda e inferior derecha
  geoSupIzq = G_NORMAL_MAP.getProjection().
    fromPixelToLatLng(xySupIzq, zoom);
  geoInfDer = G_NORMAL_MAP.getProjection().
    fromPixelToLatLng(xyInfDer, zoom);

  // genera la URL del "tile"
  urlTile  = this.wmsUrl;
  urlTile += "&REQUEST=GetMap";
  urlTile += "&SERVICE=WMS";
  urlTile += "&VERSION=" + this.wmsVersion;
  urlTile += "&LAYERS=" + this.wmsCapas;
  urlTile += "&STYLES=" + this.wmsEstilos;
  urlTile += "&FORMAT=" + this.wmsFormato;
  urlTile += "&BGCOLOR=" + this.wmsColorFondo;
  urlTile += "&TRANSPARENT=TRUE";
  urlTile += "&SRS=" + this.wmsSrs;
  urlTile += "&BBOX=" + geoSupIzq.x + "," + geoSupIzq.y + "," +
    geoInfDer.x + "," + geoInfDer.y;
  urlTile += "&WIDTH=256";
  urlTile += "&HEIGHT=256";
  urlTile += "&reaspect=false";

  return urlTile;
}
</script>
</head>
<body onload="initialize()" onunload="GUnload()">
<div id="elemento_mapa" style="width: 750px; height: 450px">
</div>
</body>
</html>
```

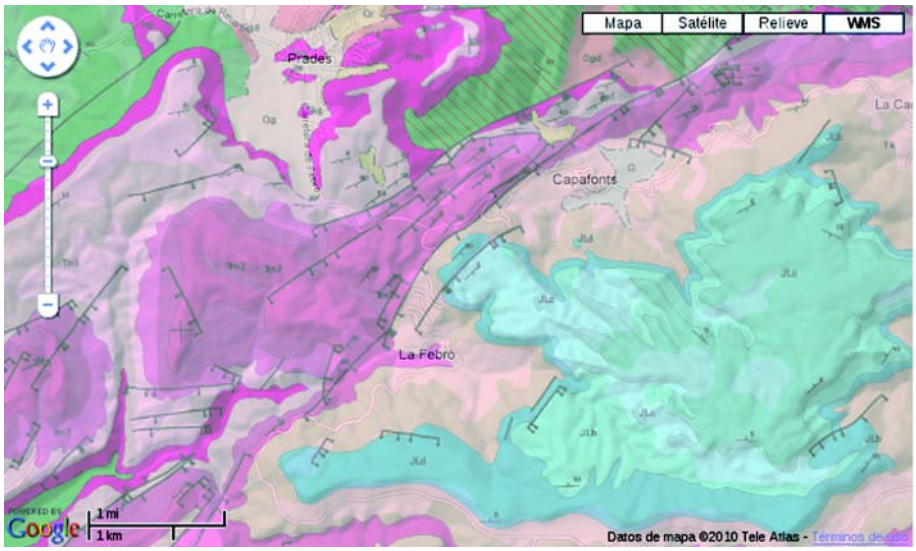
El resultado será parecido al siguiente:

Ejemplo 3: mapa WMS combinado con el mapa normal de Google Maps

En este ejemplo, hemos creado un mapa personalizado que combina el mapa normal de Google Maps con un mapa geológico a escala 1:50000 procedente del servicio WMS del ICC. Sólo mostramos la función *initialize*, porque el resto del código no cambia respecto al ejemplo n.º 2.

```
function initialize() {
  var mapa;
  var capaGeo;
  var mapaCombi;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(document.getElementById("elemento_mapa"));
    mapa.setCenter(new GLatLng(41.5580, 1.5906), 7);
    mapa.setUIToDefault();

    // crea la capa WMS
    capaGeo = crearCapaWms(7, 15, 0.4,
      "http://shagrat.icc.es/lizardtech/iserv/ows?",
      "mgc50m,");

    // crea un mapa personalizado a partir de la capa WMS y del
    // mapa normal de Google Maps
    mapaCombi = new GMapType(
      [G_NORMAL_MAP.getTileLayers()[0], capaGeo],
      G_NORMAL_MAP.getProjection(),
      "WMS");

    // agrega el mapa personalizado al visor
    mapa.addMapType(mapaCombi);
  }
}
```

El resultado será parecido al siguiente:

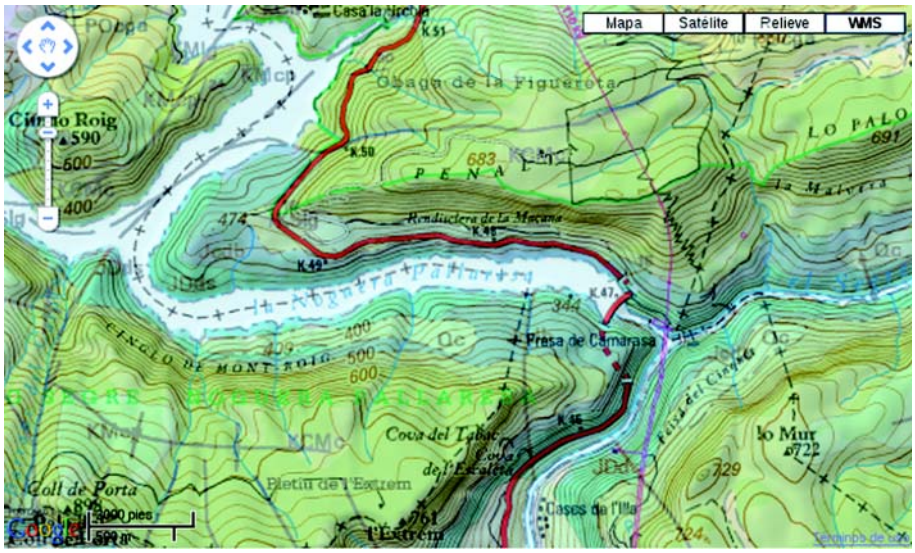


Ejemplo 4: dos mapas WMS procedentes de una única fuente combinados

El ejemplo siguiente combina dos mapas procedentes del servicio WMS del ICC: el topográfico 1:25000 y el geológico 1:50000. Sólo mostramos la función *initialize*, porque el resto del código no cambia respecto al ejemplo n.º 2.

```
function initialize() {
  var mapa;
  var capaTopo;
  var capaGeo;
  var mapaCombi;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(document.getElementById("elemento_mapa"));
    mapa.setCenter(new GLatLng(41.5580, 1.5906), 7);
    mapa.setUIToDefault();

    // crea la capa WMS del mapa topográfico
    capaTopo = crearCapaWms(7, 12, 1.0,
      "http://shagrat.icc.es/lizardtech/iserv/ows?",
      "mtc50m,");

    // crea la capa WMS del mapa geológico
    capaGeo = crearCapaWms(7, 15, 0.40,
      "http://shagrat.icc.es/lizardtech/iserv/ows?",
      "mgc50m,");

    // crea un mapa personalizado a partir de las dos capas WMS
    mapaCombi = new GmapType(
      [capaTopo, capaGeo],
      G_NORMAL_MAP.getProjection(), "WMS");

    // agrega el mapa personalizado al visor
    mapa.addMapType(mapaCombi);
  }
}
```

El resultado será parecido al siguiente:

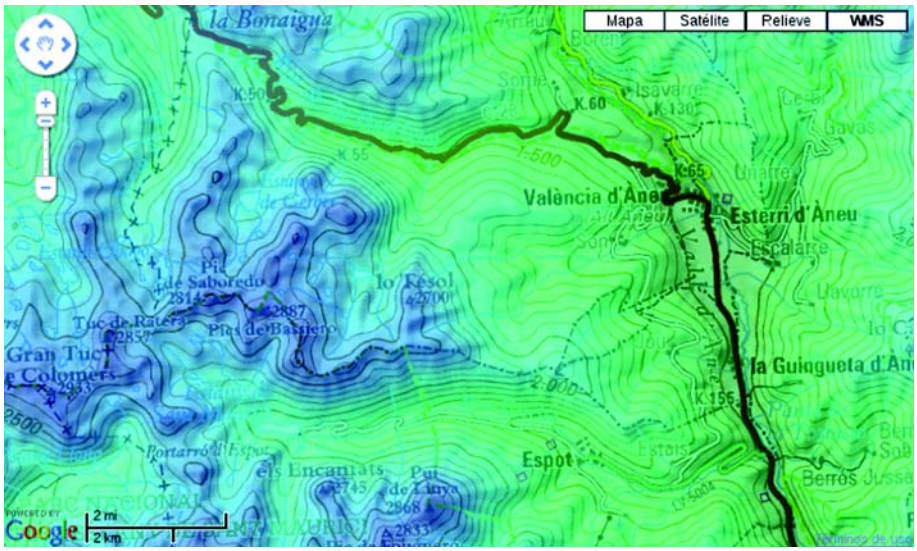


Ejemplo 5: dos mapas WMS procedentes de fuentes distintas combinados

El ejemplo siguiente combina el mapa topográfico de Cataluña a escala 1:50000 procedente del servicio WMS del ICC con el atlas climático de Cataluña (clima anual, temperatura media) procedente del servicio WMS del CREAF. Sólo mostramos la función *initialize*, porque el resto del código no cambia respecto al ejemplo n.º 2.

```
function initialize() {
  var mapa;
  var capaTopo;
  var capaClima;
  var mapaCombi;
  if (GBrowserIsCompatible()) {
    mapa = new GMap2(document.getElementById("elemento_mapa"));
    mapa.setCenter(new GLatLng(41.5580, 1.5906), 7);
    mapa.setUIToDefault();

    // crea la capa WMS del mapa del ICC
    capaTopo = crearCapaWms(7, 12, 0.5,
      "http://shagrat.icc.es/lizardtech/iserv/ows?",
      "mtc250m,");

    // crea la capa WMS del mapa del CREAF
    capaClima = crearCapaWms(7, 12, 1.0,
      "http://www.opengis.uab.es/cgi-bin/ACDC/MiraMon5_0.cgi?",
      "clima_anual-catalunya",
      "Tmit",
      "image/jpeg");

    // crea un mapa personalizado a partir de las dos capas WMS
    mapaCombi = new GMapType(
      capaClima, capaTopo],
      G_NORMAL_MAP.getProjection(),
      "WMS");

    // agrega el mapa personalizado al visor
    mapa.addMapType(mapaCombi);
  }
}
```

El resultado será parecido al siguiente:



Resumen

En este módulo hemos aprendido a ir más allá de las capacidades de Google Maps, conectándolo con bases de datos de mapas externas mediante el protocolo OGC/WMS. Esta forma de trabajar nos da acceso a una gran variedad de cartografía temática (muchas veces de carácter local) mucho más precisa que la ofrecida por Google, a la vez que permite el acceso a unos mismos datos a través de diversas fuentes, aumentando así la disponibilidad y la confiabilidad de los datos.

Para poder utilizar mapas procedentes de una fuente WMS en Google Maps, hemos aprendido, en primer lugar, a obtener mapas WMS a través de un navegador web, de la misma forma que accederíamos a cualquier sitio de la red. Una vez hecho esto, para poder visualizarlos, ha sido necesario crear mapas personalizados que posteriormente hemos agregado como una capa en alguno de los tipos de mapa predeterminados de Google Maps (mapa, satélite o relieve), o en uno de nuevo.

Al final, hemos aprendido a combinar mapas de Google Maps, con mapas WMS, e incluso mapas WMS procedentes de diferentes fuentes.

Bibliografía

Institut Cartogràfic de Catalunya, "*Exemple bàsic Google Maps amb capes ICC*", http://www.icc.es/cat/content/download/9942/32328/file/gm_wms_helloworld.zip.

Just van den Broecke, "*Google Maps API Experiments*", http://www.geoskating.com/gmap/.

Carlos Granell Canut, "*Servicios OGC*", FUOC 2009.

Google, "*Google Maps JavaScript API V2 Reference*", http://code.google.com/intl/es/apis/maps/documentation/javascript/v2/reference.html.

Google, "*Google Maps API Concepts*", http://code.google.com/intl/es/apis/maps/documentation/javascript/v2/basics.html.

Open GIS Consortium Inc., "*Web Map Service Implementation Specification*", http://portal.opengeospatial.org/files/?artifact_id=1081&version=1&format=pdf, version 1.1.1, 16/01/2002.